プログラミングおよび実習II・中野・水 3 講時・定期試験問題 2004 年 1 月 28 日

注意 問題は 2 問あります。 解答はすべて別紙の解答用紙に記入しなさい。この試験の採点結果は、 2 月 4 日 (水) までに計算機実習室 (Linux 環境) の `mprog2` コマンドで表示されるようにする予定です。

I カレントディレクトリに置かれた `PriceList` という名前のファイルに、いくつかの店舗で調査した野菜の価格に関するデータが次のような形式で格納されている。

PriceList (一部省略)

```
Midoriya Daikon 314
Akariya Komatuna 192
Midoriya Shoga 167
Hinodeya Shoga 154
Akariya Nira 281
Yaohachi Hakusai 171
Midoriya Piman 201
 ⋮
Akariya Hakusai 114
Yaohachi Nira 178
Hinodeya Satsumaimo 371
```

このファイルの各行は、

店舗名 商品名 価格

の形式になっており、これら 3 つの部分はスペースで区切られている。店舗名や商品名はすべてローマ字 (先頭のみ大文字) で記述されており、どれも 80 文字未満となっている。

次のプログラム `average1.c` は、この `PriceList` というファイルからすべてのデータを読み込んだ後、標準入力 (キーボード) から入力された商品名について、その平均価格を (小数点以下 1 桁まで) 出力するプログラムである。以下の実行例と実現の方針を参考にして、このプログラムで省略されている部分を補いなさい。

average1.c (一部省略)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define FILE_NAME        "PriceList"
#define MAX_DATA        (10000)
#define NAME_LEN        (80)

typedef struct {
    char store[NAME_LEN];       /* 店舗名 */
    char vegie[NAME_LEN];       /* 商品名 */
    int price;                  /* 価格 */
} Item;
```

(1) 関数 average1 の定義

```
int readfile(Item d[], char *file)
{
    FILE *fp;
    int num;

    fp = fopen(file, "r");
    if (fp == NULL) {
        printf("%s というファイルが読めません\n", file);
        exit(EXIT_FAILURE);
```

```
    }

    (2) ファイル中のデータをすべて配列 d に読み込むためのプログラム

    fclose(fp);
    return num;
}

int main()
{
    static Item data[MAX_DATA];
    int num;
    char name[NAME_LEN];
    double avr;

    num = readfile(data, FILE_NAME);
    while (1) {
        printf ("商品名を入力してください : ");
        if (scanf("%s", name) != 1 || name[0] == '.')
            break;
        avr = average1(data, num, name);
        if (avr < 0.0)
            printf("その商品に関するデータがありません。\n");
        else
            printf("平均価格 %.1f円\n", avr);
    }
    return EXIT_SUCCESS;
}
```

average1.c のコンパイルと実行例

```
s1542h017% cc -o average1 average1.c
s1542h017% ./average1
商品名を入力してください : Daikon
平均価格 282.2円
商品名を入力してください : Ninjin
平均価格 146.8円
商品名を入力してください : Banana
その商品に関するデータがありません。
商品名を入力してください : .
s1542h017%
```

関数 average1 の実現の方針

(1) Item 型の配列 d、要素数 n、商品名 name の 3 つを引数として受け取る。

(2) 配列 d の先頭から n 個の要素の内、商品名が name であるようなものの個数と価格の総和を求める。このためには、これまでに見つけた要素の個数と、その価格の総和を記憶するための変数を用意しておき、配列 d の先頭から、線形探索と同様の手順で商品名が name である要素を探し、そのような要素が見つかるたびに見つけた個数や価格の総和を更新していけばよい。2 つの文字列の比較には、標準ライブラリ関数 strcmp を用いる。関数 strcmp は、2 つの文字列を char * 型の引数で受け取り、文字コードでの辞書式順で、第 1 引数の方が第 2 引数の方より先である場合は負の整数を、後である場合は正の整数を、2 つが同じ文字列である場合は 0 を (int 型の) 返り値として返す。

(3) 求めた総和を個数で割って平均価格を算出し、結果を double 型の返り値として返す。一つも見つからなかった場合は -1 を返す。

II 次のプログラム average2.c は、問題 I の average1.c と同じことを別の方法で行なうプログラムである。各関数の実現の方針に従って、省略されている部分のプログラムを補ってこのプログラムを完成させなさい。ただし、関数 readfile の定義は、問題 I のものをそのまま使用するものとし、解答用紙にその定義を書く必要はありません。

average2.c（一部省略）

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define FILE_NAME        "PriceList"
#define MAX_DATA        (10000)
#define NAME_LEN        (80)

typedef struct {
    char store[NAME_LEN];       /* 店舗名 */
    char vegie[NAME_LEN];       /* 商品名 */
    int price;                  /* 価格 */
} Item;
```

（1）関数 quicksort の定義

（2）関数 binarysearch の定義

（3）関数 average2 の定義

```
int readfile(Item d[], char *file)
{
```

問題I average1.c の関数 readfile と同じ

```
}

int main()
{
    static Item data[MAX_DATA], *dp[MAX_DATA];
    int i, num;
    char name[NAME_LEN];
    double avr;

    num = readfile(data, FILE_NAME);
    for (i = 0; i < num; i++)
        dp[i] = &data[i];
    quicksort(dp, 0, num-1);
    while (1) {
        printf ("商品名を入力してください : ");
        if (scanf("%s", name) != 1 || name[0] == '.')
            break;
        avr = average2(dp, num, name);
        if (avr < 0.0)
            printf("その商品に関するデータがありません。\n");
        else
            printf("平均価格 %.1f円\n", avr);
    }
    return EXIT_SUCCESS;
}
```

プログラム全体の実現の方針

(1) 読み込んだデータは商品名順になるように整列しておく。

(2) 標準入力 (キーボード) から入力された商品名について、まず 2 分探索法で整列されたデータの中から探し出す。

(3) 見つかった要素を基準として (整列されたデータの中で) 前方と後方のそれぞれに向って同じ商品名のデータが続く範囲を調べ、その範囲のデータの件数と価格の総計を求めて平均価格を算出する。

関数 quicksort の実現の方針

(1) Item 型へのポインタ型の配列、整列を行なう範囲の先頭要素の添字、同じく最後尾の要素の添字の 3 つを引数として受け取る。

(2) 指定された範囲の配列要素を、各要素が指す構造体 (Item 型) の vegie の部分 (メンバ) が文字コードでの辞書式順になるようにクイックソート法で整列させる。文字列の比較は標準ライブラリ関数 strcmp を使って行なう。

関数 binarysearch の実現の方針

(1) Item 型へのポインタ型の配列、配列中の要素数、探したい商品名 (char * 型) の 3 つを引数として受け取る。

(2) 配列中の要素から、要素が指す構造体 (Item 型) の vegie の部分 (メンバ) が (引数で指定された) 商品名と同じであるものを 2 分探索法で探し、最初に見つかった要素の添字を返り値 (int 型) として返す。見つからなかった場合は -1 を返す。文字列の比較には、標準ライブラリ関数 strcmp を用いる。

関数 average2 の実現の方針

(1) Item 型へのポインタ型の配列 d、配列中の要素数 n、平均価格を知りたい商品名 name (char * 型) の 3 つを引数として受け取る。

(2) まず、関数 binarysearch を呼び出して、name で指定された商品のデータを配列 d の中から探索し、見つかった添字を int 型の変数 (例えば k) に格納する。

(3) その商品に関するデータの個数と価格の総和を記憶するための変数 (例えば、それぞれ count と total) を用意しておき、それぞれ、 1 と配列 d の添字 k の要素が指す価格で初期化する。

(4) 添字 k-1 から始めて、配列 d の先頭に向って各要素が指すデータを調べ、指定された商品名が続く限り、その価格を total に足し込むとともに、変数 count の値を 1 ずつ増やしていく。配列の先頭要素を調べ終るか、異る商品名のデータ (を指す要素) に出会った時に、この処理は終了する。

(5) 同様に、添字 k+1 から始めて、配列 d の末尾に向って各要素が指すデータを調べ、指定された商品名が続く限り、その価格を total に足し込むとともに、変数 count の値を 1 ずつ増やしていく。配列の最後尾 (添字 n-1) の要素を調べ終るか、異る商品名のデータ (を指す要素) に出会った時に、この処理は終了する。

(6) 以上の処理をすべて終えたら、total の値と count の値から、その商品の平均価格を算出し、結果を double 型の返り値として返す。

関数 main の実現の方針

(1) Item 型へのポインタ型の配列 dp を用意しておき、ファイル中のデータを Item 型の配列 data にすべて読み込んだ後、配列 dp の各要素を data 中の対応する要素のアドレスで初期化する。

(2) 関数 quicksort を呼び出して、配列 dp 中の要素を、それが指す構造体 (Item 型) の vegie の部分 (メンバ) が文字コードでの辞書式順となるように整列させる。

(3) 標準入力から商品名が入力されるたびに、関数 average2 を呼び出して得られた平均価格を出力する。

(定期試験問題終り)

プログラミングおよび実習II(中野クラス)・定期試験・解答例

I (1)

```
double average1(Item d[], int n, char *name)
{
    double total = 0.0;
    int count = 0, i;

    for (i = 0; i < n; i++) {
        if (strcmp(d[i].vegie, name) == 0) {
            total += d[i].price;
            count++;
        }
    }

    if (count == 0)
        return -1.0;

    return total/count;
}
```

I (2)

```
    num = 0;
    while (fscanf(fp, "%s %s %d",
                  d[num].store, d[num].vegie, &d[num].price) == 3) {
        num++;
    }
```

II (1)

```
void quicksort(Item *d[], int h, int t)
{
    Item *tmp;
    int i, j;
    char *r;

    if (h >= t)
        return;

    r = d[(h+t)/2]->vegie;
    i = h;
    j = t;
    while (1) {
        while (strcmp(d[i]->vegie, r) < 0)
            i++;
        while (strcmp(r, d[j]->vegie) < 0)
            j--;
        if (j <= i)
            break;
        tmp = d[i];
        d[i++] = d[j];
        d[j--] = tmp;
    }
    quicksort(d, h, i-1);
    quicksort(d, j+1, t);
}
```

II (2)

```
int binarysearch(Item *d[], int n, char *name)
{
    int h = 0, t = n-1, m;

    while (h <= t) {
        m = (h+t)/2;
        if (strcmp(name, d[m]->vegie) < 0)
            t = m-1;
        else if (strcmp(d[m]->vegie, name) < 0)
            h = m+1;
        else
            return m;
    }
    return -1;
}
```

II (3)

```
double average2(Item *d[], int n, char *name)
{
    int start, count, i;
    double total;

    start = binarysearch(d, n, name);

    if (start < 0)
        return -1.0;

    total = d[start]->price;
    count = 1;
    for (i = start-1; i >= 0; i--) {
        if (strcmp(d[i]->vegie, name) != 0)
            break;
        total += d[i]->price;
        count++;
    }
    for (i = start+1; i < n; i++) {
        if (strcmp(d[i]->vegie, name) != 0)
            break;
        total += d[i]->price;
        count++;
    }
    return total/count;
}
```

(解答例終り)